AD-A035 917 TEXAS UNIV AT AUSTIN CENTER FOR CYBERNETIC STUDIES F/G 12/2
SOLVING SINGULARLY CONSTRAINED GENERALIZED NETWORK PROBLEMS.(U)
SEP 76 J HULTZ, D KLINGMAN N00014-75-C-0616
UNCLASSIFIED CCS-256 NL

1 OF 1
AD-A
035 917

END
DATE
FILMED
3-25-77
NTIS

SOLVING SINGULARLY CONSTRAINED GENERALIZED NETWORK PROBLEMS

TEXAS UNIVERSITY AT AUSTIN

SEPTEMBER 1976

CENTER FOR CYBERNETIC STUDIES

The University of Texas
Austin, Texas 78712

Research Report CCS 256

SOLVING SINGULARLY CONSTRAINED GENERALIZED NETWORK PROBLEMS

by

John Hultz
D. Klingman

September 1976

This research was partly supported by ONR Contract N00014-76-C-0383 with Decision Analysis and Research Institute and by Project NR047-021, ONR Contracts N00014-75-C-0616 and N00014-75-C-0569 with the Center for Cybernetic Studies, The University of Texas. Reproduction in whole or in part is permitted for any purpose of the United States Government.

CENTER FOR CYBERNETIC STUDIES

A. Charnes, Director
Business-Economics Building, 203E
The University of Texas
Austin, Texas 78712
(512) 471-1821

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Center for Cybernetic Studies The University of Texas	Unclassified
	2b. GROUP

3. REPORT TITLE

"Solving Singularly Constrained Generalized Network Problems"

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

5. AUTHOR(S) (First name, middle initial, last name)

John Hultz
Darwin Klingman

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
September 1976	36	25

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
N00014-76-C-0383 & N00014-75-C-0569; 0616	Center for Cybernetic Studies Research Report CCS 256
b. PROJECT NO. NR047-021	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Office of Naval Research (Code 434) Washington, D.C.

13. ABSTRACT

The singularly constrained generalized network problem represents a large class of capacitated linear programming (LP) problems. This class includes any LP problem whose coefficient matrix, ignoring single upper bound constraints, contains m + 1 rows which may be ordered such that each column has at most two non-zero entries in the first m rows. The paper describes efficient procedures for solving such problems and presents computational results which indicate that procedures are at least five times faster than the state of the art LP systems MPS-360 and APEX-III.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6811

Unclassified

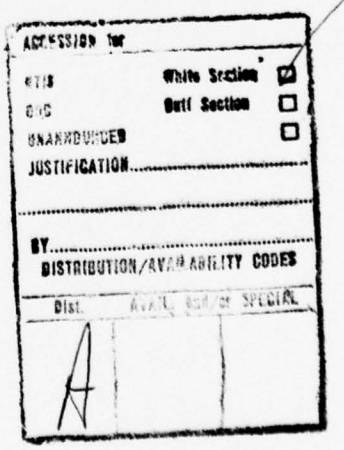
Security Classification

A-31408

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Networks						
Weighted Networks						
Generalized Networks						
Transportation Problems						
Linear Programming Problems						

ABSTRACT

The singularly constrained generalized network problem represents a large class of capacitated linear programming (LP) problems. This class includes any LP problem whose coefficient matrix, ignoring single upper bound constraints, contains m + 1 rows which may be ordered such that each column has at most two non-zero entries in the first m rows. The paper describes efficient procedures for solving such problems and presents computational results which indicate that procedures are at least five times faster than the state of the art LP systems MPS-360 and APEX-III.



1.0 INTRODUCTION

The singularly constrained generalized network problem represents a large class of capacitated linear programming (LP) problems. This class includes any LP problem whose coefficient matrix, ignoring simple upper bound constraints, contains m + 1 rows which may be ordered such that each column has at most two non-zero entries in the first m rows. There are numerous real world applications of problems containing this structure. For example, a major automobile manufacturer has devised a singularly constrained generalized network model in which the extra constraint is due to tariff regulations that dollar flows across the U.S.-Canadian border be equal. Another example is a cash flow model [4] used by the U.S. Treasury Department to analyze the effect of tax regulations on multinational firms in which the extra constraint is used to balance direct loans.

A large portion of the literature on LP problems has been devoted to special cases of the singularly constrained generalized network problem. In particular, if the m + 1st row of the coefficient matrix is a zero vector, then depending on the structure of the first m rows, the problem is either a shortest path, assignment, transportation, transshipment, generalized transportation, or generalized transshipment problem. The most efficient procedures for solving these specializations are based on viewing the problems in a graphical context. In particular, the simplex algorithm has been adapted [1, 2, 7, 10, 14, 15, 18, 20, 21, 22, 25] to solve problems in which the coefficient matrix and the basis matrix are stored as graphs using computer list structures.

The use of such structures reduces both the amount of work needed to perform the basic simplex steps and the amount of computer memory required to store essential data. The matrix operations of finding the representation of an entering vector and determining updated pseudo dual variable values are performed by tracing paths within the basis graph. Since the graphs contain only the non-zero entries in the problem (basis), the list procedures [15, 23, 24] eliminate checking or performing unnecessary arithmetic operations on zero elements. In addition, by exploiting the triangular or near-triangular properties of such problems, these procedures allow the basis matrix inverse to be stored implicitly as a basis graph. This graph is updated during basis exchange steps by simply changing a few pointers in the list structures. This entirely eliminates the arithmetic operations normally used to update the basis inverse and also eliminates possible round-off error.

The advantages of such procedures are dramatically illustrated by the fact that such simplex codes [8] for solving capacitated transshipment problems which use graphical processing techniques [7] are at least 75 times faster than state of the art commercial LP codes. In fact, a graphical processing code for solving transportation problems [9] has been shown to be 150 times faster than the OPHELIE LP code.

In addition to improving solution speed, the network processing techniques have the added advantage of requiring less computer memory to solve a problem. This allows larger problems to be solved without resorting to slower external storage procedures. If the use of external storage is unavoidable, the graphical procedures are normally able to keep the entire

basis in central memory and to facilitate efficient input/output processing of non-basic variable data.

Motivated by these advantages, this paper develops efficient procedures for solving singularly constrained generalized network problems via the simplex method. These procedures exploit the topological structure of the first m rows of the coefficient matrix. In particular, the working basis is partitioned so that the near triangularity property of the basis can be fully exploited from both the algebraic and graphical processing viewpoints.

The importance of this research is actually twofold. First, it has resulted in a highly efficient algorithm and computer code. Computational testing of the proposed algorithm indicates that it is five times more efficient than the general purpose LP codes, MPS-360 and APEX-III. Second, it focuses attention of the importance and practicality of viewing many LP problems as being network related, especially during the formulation stages.

2.0 PROBLEM DEFINITION

A singularly constrained generalized network problem is defined as follows:

$$\text{Minimize} \quad c_1^T x + c_2^T y \tag{1}$$

subject to:

$$Gx + Oy = b \tag{2}$$

$$f_1 x + f_2 y = f_0 \tag{3}$$

$$0 \leq x \leq u \tag{4}$$

$$0 \leq y \leq v \tag{5}$$

where each column of G contains at most two non-zero entries. The constraints (2) will henceforth be referred to as the underlying generalized network problem and equation (3) as the extra constraint.

It is often useful in the process of formulating and solving unconstrained generalized network problems to view them in a general graph structure [6]. A general graph is an extension of a simple graph G(V, E), where V is a finite non-empty set of vertices and E is a finite set of edges, each of which is identified with an unordered pair of distinct elements of V. Each edge of E connects two vertices which are then considered adjacent. If E is expanded to contain edges that have both endpoints incident on the same vertex (loops) or multiple edges connecting the same two vertices (parallel edges) then G(V, E) is called a general graph.

The underlying generalized network of the problem (1-5) forms such a general graph. Each column of G having two non-zero entries relates to an edge connecting two vertices. Each column containing a single non-zero entry forms an edge with both endpoints incident on the same vertex. Such edges will be referred to as loops. Associated with each edge is a variable, an upper bound, a cost coefficient, and two non-zero multipliers (one in the case of loops). The multipliers govern how much of the variable is to be added to or subtracted from the appropriate vertex values (right-hand side values).

Most of the literature on generalized network problems assumes that each edge has exactly two non-zero multipliers which are of opposing sign. Further, it is assumed without loss of generality that one of the multipliers has a value of -1. For this special class of generalized network problems a direction may be imputed to each edge. The graph is then called a digraph and the variables are customarily considered to be flows across the directed edges or arcs.

If artificial variables are appended to the problem, it may be assumed without loss of generality that the underlying generalized network problem contains a coefficient matrix which has full row rank. If the coefficient matrix does not have full row rank, it is possible to further simplify the operations presented subsequently based on the fact that any generalized network without full row rank is equivalent to a disjoint set of minimum cost flow networks [11].

3.0 BASIS STRUCTURE

Let m be the number of rows in G. Using the standard bounded variable simplex algorithm, a basis for the singularly constrained generalized network problem will be a linearly independent set of m + 1 column vectors taken from the coefficient matrix corresponding to constraints (2) and (4). Let $\bar{P}_i = \begin{bmatrix} P_i \\ f_i \end{bmatrix}$, $i = 1, 2, \ldots, m, m+1$ denote these basis vectors, where P_i represents the first m components and f_i the last component of $\bar{P}_i$. It is important to observe that the rank of $S = \{P_1, P_2, \ldots, P_m, P_{m+1}\}$ is exactly m. For notational convenience, assume the set consisting of the first m vectors of S in linearly independent. Letting $B_n = [P_1, P_2, \ldots, P_m]$ and $F_n = [f_1, f_2, \ldots, f_m]$, any basis for the problem (1-5) may be partitioned as follows:

$$B = \begin{bmatrix} B_n & P_{m+1} \\ F_n & f_{m+1} \end{bmatrix} . \qquad (6)$$

Since B_n is non-singular and is composed of vectors taken from (2), it forms a working basis for the underlying generalized network problem. The graphical structure of B_n has been characterized in the literature [5, 13, 21, 22]. It contains all of the vertices but only a subset of the edges of the original general graph, and, hence, is called a spanning subgraph. Due to the

non-singularity of B_n, the spanning subgraph is simply a finite set of disjoint quasi-trees. Each quasi-tree is a simple tree to which a single edge has been added. The additional edge creates exactly one cycle (a circular path) in the quasi-tree. It should be noted that a loop, as defined previously, is a cycle involving a single vertex and a single edge.

An efficient method for storing the general graph B_n has been developed, called the extended augmented predecessor index (EAPI) method [13]. The EAPI method, based on the triple-label method of Johnson [16, 17], employs computer list structures to represent the vertices and edges of B_n. These lists provide the means for performing the two types of quasi-tree traversal needed to execute the basic simplex steps. Each quasi-tree may be conceptually oriented with the cycle at the top and all attached subtrees hanging down from this "rooted cycle". Representations of entering edges are found by tracing paths from given vertices up to and around the associated cycles. Pseudo dual variable values are updated by locating all vertices situated "below" a given vertex. The efficiency of the EAPI method derives from the facts that only non-zero basis coefficients are stored, that operations involving a basis matrix inverse may be performed using the basis graph, and that basis exchange steps involve only the updating of list structure pointers.

The simplex procedures in the following sections are based on the basis partitioning of (6). The efficiency of these procedures can best be realized if B_n is stored via the EAPI method. Therefore, each operation will correspondingly be described in terms of the two quasi-tree traversal methods briefly described earlier. However, a thorough knowledge of the EAPI method will not be required to understand the mathematical operations involved in performing the basic simplex steps.

4.0 OPTIMALITY TESTING

Once a basis has been selected for any linear programming problem, the simplex method dictates that it be checked for optimality by computing updated objective function coefficients. For network related problems, as with general linear programming problems, an efficient method for performing this operation is to use complementary slackness to calculate pseudo dual variable values associated with the current primal basis and determine if this dual solution is feasible. In other words, once pseudo dual variable values have been determined, an updated objective function coefficient is equivalent to the difference between the left and right hand sides of the associated dual constraint.

The dual problem associated with the singularly constrained generalized network problem (1-5) is:

$$\text{Maximize} \quad w^T b + Sf_0 - \gamma^T u - \psi^T v \tag{7}$$

subject to:

$$w^T G + Sf_1 - \gamma^T \leq c_1^T \tag{8}$$

$$Sf_2 - \psi^T \leq c_2^T \tag{9}$$

w, S unrestricted

$\gamma, \psi \geq 0$.

Pseudo dual variable values w and S associated with the current primal working basis B may be found using the theorem of complementary slackness, which yields the following system of equations:

$$w^T B_n + SF_n = c_n^T \tag{10}$$

$$w^T P_{m+1} + Sf_{m+1} = c_{m+1} \tag{11}$$

where (c_n^T , c_{m+1}) is an appropriately ordered vector of the objective function coefficients associated with the column vectors of B.

If the value of S in (10) is known, w can be computed using

$$w^T B_n = c_n^T - SF_n . \qquad (12)$$

Assuming B_n is stored via the EAPI method, (12) may be solved using an extension of the "pricing-out" procedure for generalized networks. This method associates each variable in w with a vertex in the general graph B_n. A pseudo dual variable value associated with a single cycle vertex of a quasi-tree is determined using the "cycle factor" [12, 13]. The remaining pseudo dual variable values in the quasi-tree are determined by exploiting the near triangularity of the system of equations (12) and using the downward traversal capabilities of the EAPI method. Note that if a basis exchange has just occurred and S has not changed from its previous value, the maximum number of pseudo dual variable values that have to be updated will be those associated with a single quasi-tree. This particular instance, which is standard in generalized networks, is due to the deletion and insertion of a single equation in the disjoint, near triangular system (12).

The solution of (12) is dependent upon knowing the current value of S. Its value may be determined by combining (11) and (12) above and performing the following simple algebraic manipulation:

$$(c_n^T B_n^{-1} - SF_n B_n^{-1})P_{m+1} + Sf_{m+1} = c_{m+1}$$

$$S(F_n B_n^{-1} P_{m+1} - f_{m+1}) = c_n^T B_n^{-1} P_{m+1} - c_{m+1}$$

$$S = (c_n^T B_n^{-1} P_{m+1} - c_{m+1})/(F_n B_n^{-1} P_{m+1} - f_{m+1}). \qquad (13)$$

The vector $B_n^{-1}P_{m+1}$, used twice in (13), is simply the representation of P_{m+1} in terms of the current basis B_n for the underlying generalized network problem. In other words, it is a vector whose components, α_i, $i = 1, 2, \ldots, m$, satisfy:

$$\alpha_1 P_1 + \alpha_2 P_2 + \ldots + \alpha_m P_m = P_{m+1}. \tag{14}$$

Using this fact, equation (13) may be restated as

$$S = (\sum_{i=1}^{m} \alpha_i c_i - c_{m+1})/(\sum_{i=1}^{m} \alpha_i f_i - f_{m+1}). \tag{15}$$

Computationally, S may be found by tracing paths in the general graph B_n. P_{m+1} is related to an edge, not in B_n, that is incident on at most two vertices. Using the EAPI method, the unique paths from these vertices to their respective cycle(s) may be found. These paths in conjunction with the non-repeating path(s) around the associated cycle(s) generate only the non-zero values of the α_i, $i = 1, 2, \ldots, m$ (See [13]). As these values are determined, the terms

$$\theta = \sum_{i=1}^{m} \alpha_i f_i - f_{m+1} \tag{16}$$

and

$$\rho = \sum_{i=1}^{m} \alpha_i c_i - c_{m+1} \tag{17}$$

may be easily computed. Upon completion, S is determined as $S = \rho/\theta$. Note that if $\bar{P}_{m+1}$ is associated with a y_k variable (i.e., P_{m+1} is the zero vector), each $\alpha_i = 0$, $i = 1, 2, \ldots, m$. No graph traversal is needed in this case and $S = c_{m+1}/f_{m+1}$.

Once pseudo dual variable values have been determined, updated objective function coefficients may be calculated as follows:

$$\bar{c}_i = [\, w^T, S] \begin{bmatrix} P_i \\ f_i \end{bmatrix} - c_i \qquad \forall\, i \tag{18}$$

or

$$\bar{c}_i = w^T P_i + S f_i - c_i \quad , \quad \forall\, i. \tag{19}$$

Dual, and hence primal, optimality is achieved if the following conditions hold for all non basic variables:

$$\bar{c}_i \leq 0, \; x_i = 0 \text{ , or } y_i = 0$$

$$\bar{c}_i = 0, \; x_i \text{ o} \; y_i \text{ - basic}$$

$$\bar{c}_i \geq 0, \; x_i = u_i \text{ or } y_i = v_i, \qquad \forall\, i \,. \tag{20}$$

5.0 FINDING THE REPRESENTATION OF A VECTOR TO ENTER THE BASIS

If the current basis is non-optimal (i.e., at least one of the conditions (20) is violated), then a pivot eligible vector is selected and a basis exchange step is performed. To perform this basis exchange requires the determination of the representation $\bar{Z}_r = \begin{pmatrix} Z_r \\ z_r \end{pmatrix}$ of the entering vector $\bar{P}_r = \begin{pmatrix} P_r \\ f_r \end{pmatrix}$ in terms of the current basis B. This representation may be found by solving the following system of equations:

$$B_n Z_r + P_{m+1} z_r = P_r \tag{21}$$

$$F_n Z_r + f_{m+1} z_r = f_r \,. \tag{22}$$

Employing what is essentially the double-reverse method of Charnes and Cooper [3], (21) may be written as

$$Z_r = B_n^{-1} P_r - B_n^{-1} P_{m+1} z_r \,. \tag{23}$$

Substituting (23) into (22) and applying algebraic manipulation yields the following:

$$F_n(B_n^{-1}P_r - B_n^{-1}P_{m+1}z_r) + f_{m+1}z_r = f_r$$

$$(f_{m+1} - F_nB_n^{-1}P_{m+1})z_r = f_r - F_nB_m^{-1}P_r$$

$$z_r = (F_nB_n^{-1}P_r - f_r)/(F_nB_n^{-1}P_{m+1} - f_{m+1}). \qquad (24)$$

The vector $B_n^{-1}P_r$ in the numerator of (24) is the representation of P_r in terms of the basis B_n for the underlying generalized network problem. Letting β_i, $i = 1, 2, \ldots, m$ denote the components of this vector yields:

$$\beta_1P_1 + \beta_2P_2 + \ldots + \beta_mP_m = P_r. \qquad (25)$$

Noting that the denominator of (25) is exactly θ determined in (16), (24) may be stated as

$$z_r = (\sum_{i=1}^{m} \beta_if_i - f_r)/\theta. \qquad (26)$$

Computationally, the EAPI method may be employed to efficiently determine the non-zero β_i, $i = 1, 2, \ldots, m$ of (25). This is directly analogous to the procedure described in section 4 for finding the representation of P_{m+1} in terms of B_n. As the β_i are determined, the numerator of (26) may be successively summed. Upon completion, the previously obtained value of θ may be used to determine z_r.

Using the value of z_r as found in (26) and the representation of P_r and P_{m+1} in terms of B_n, (23) may be stated in vector form as

$$Z_r = \begin{pmatrix} \beta_1 - \alpha_1 z_r \\ \beta_2 - \alpha_2 z_r \\ \vdots \\ \beta_m - \alpha_m z_r \end{pmatrix} . \qquad (27)$$

The efficiency of (27) derives from the fact that all required information has been generated in previous operations. There are, however, certain cases in which the computations involved in determining $\bar{Z}$ may be further reduced.

In particular, if $\bar{P}_r$ is associated with a y_k, $k \in K$, then $\beta_i = 0$, $i = 1, 2, \ldots, m$. Thus (26) reduces to

$$z_r = -f_r / \theta$$

and (27) becomes

$$Z_r = f_r \begin{pmatrix} \alpha_1 / \theta \\ \alpha_2 / \theta \\ \vdots \\ \alpha_m / \theta \end{pmatrix}$$

On the other hand, $\bar{P}_{m+1}$ may be associated with a y_k, $k \in K$, in which case, $\alpha_i = 0$, $i = 1, 2, \ldots, m$,

$$z_r = (f_r - \sum_{i=1}^{m} \beta_i f_i) / f_{m+1}$$

and

$$Z_r = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{pmatrix}$$

6.0 MAINTAINING A PARTITIONED BASIS

Once the representation of the vector entering the basis has been found, a bounded variable ratio test may be performed to determine the vector to leave the basis. This involves using the representation, the current variable values, and the upper bounds. Computationally it is advantageous to couple this step with the operations performed in section 5 since only the non-zero entries of $\bar{Z}_r$ need to be checked. The basis exchange step must not be performed indiscriminately, however, in order to preserve the basis partitioning. The partitioning of the basis is the foundation upon which these procedures are built. The destruction of that partitioning would necessarily invalidate the given solution procedures.

There are three possible cases which may occur depending upon the vector selected to leave the basis.

Case 1: The vector to leave the basis is $\bar{P}_{m+1}$. In this case, the pivot may be performed directly. The underlying basis graph B_n will remain unchanged. Note that the current β_i, $i = 1, 2, \ldots, m$ values may be used as the future α_i, $i = 1, 2, \ldots, m$ values in the computation of S and in the next pivot operation.

Case 2: The vector to leave the basis is $\bar{P}_j \neq \bar{P}_{m+1}$ and $\beta_j = 0$. If the basis exchange step is performed immediately, B_u would no longer be a basis for the underlying generalized network and the basis partitioning would be destroyed. In other words, $\{P_1, P_2, \ldots, P_{j-1}, P_{j+1}, \ldots, P_m, P_r\}$ would not be a linearly independent set. However, it must be true that $\{P_1, P_2, \ldots, P_{j-1}, P_{j+1}, \ldots, P_m, P_{m+1}\}$ is a linearly independent set. The

partitioning may, thus, be preserved by switching $\bar{P}_j$ and $\bar{P}_{m+1}$. After performing this exchange, the pivot may be performed as in case 1.

Case 3: The vector to leave the basis is $\bar{P}_j \neq \tilde{P}_{m+1}$ and $\beta_j \neq 0$. The underlying basis graph B_n may be modified by the removal of P_j and the insertion of P_r. Note that if $\alpha_j = 0$ in this case, each α_i, $i = 1, 2, \ldots, m$ and S will remain unchanged during the basis exchange. Since S does not change, the efficient procedures presented in section 4 may be used to update (as opposed to recalculate) the remaining dual variable values.

In both cases 2 and 3 above, a change of pointers in the computer list structures will be necessary. This is fully described in [13].

7.0 NUMERICAL EXAMPLE

Consider the following singularly constrained generalized network problem:

Minimize $1x_{12}+6x_{13}+5x_{24}+8x_{25}+4x_{23}+4x_{34}+4x_{43}+7x_{45}+6x_{46}+0y_1+My_2$

subject to

$$
\begin{aligned}
1x_{12}+2x_{13} &= 15\\
1x_{12} - 1x_{24} - 4x_{25} + 1x_{23} &= -10\\
1x_{13} - 1x_{23} - 4x_{34} + 1x_{43} &= 0\\
-\tfrac{1}{4}x_{24} - 3x_{34} + 1x_{43} + 1x_{45} + 3x_{46} &= 0\\
2x_{25} + \tfrac{3}{4}x_{45} &= 5\\
3x_{46} &= 5\\
1x_{12} + 1x_{23} + 1x_{45} - 1y_1 + 1y_2 &= 2
\end{aligned}
$$

$$x_{ij} \geq 0$$

$$y_1, y_2 \geq 0$$

where y_2 is an artificial variable.

An optimal basis for the underlying generalized network problem is composed of the vectors P_{13}, P_{24}, P_{25}, P_{23}, P_{34}, and P_{46}. Adding to these the vector associated with y_2 and expanding yields a basis for the constrained problem

$$B = \begin{pmatrix} B_n & P_{m+1} \\ F_n & f_{m+1} \end{pmatrix} =$$

$$\left(\begin{array}{cccccc|c} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -4 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & -4 & 0 & 0 \\ 0 & -\frac{1}{4} & 0 & 0 & -3 & 3 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right)$$

The graph corresponding to B_n is shown in Figure 1. The numbers at the endpoints of each edge are the corresponding non-zero multipliers. Not indicated on the graph is $y_2 = \frac{3}{4}$.

The first step of the algorithm is to compute a value for S. Letting P_{y_2} be the vector associated with y_2, (15) reduces to $S = M$.

The system of equations associated with (12) is

$$2w_1 + 1w_3 = 6 \tag{29}$$

$$-1w_2 - \frac{1}{4}w_4 = 5 \tag{30}$$

$$-4w_2 + 2w_5 = 8 \tag{31}$$

$$1w_2 - 1w_3 = 4 - 1M \tag{32}$$

$$-4w_3 - 3w_4 = 4 \tag{33}$$

$$3w_4 + 3w_6 = 6 \tag{34}$$

Concentrating on the equations involved in the cycle (i.e., (30), (33), and (32)) the method in [13] may be used to find $w_2 = (18 - M)/(1-3) = -9 + \frac{1}{2}M$. One pass through the graph determines the other dual variable values as $w_1 = 19/2 - 3/4M$, $w_3 = -13 + 3/2M$, $w_4 = 16 - 2M$, $w_5 = -14 + M$, and $w_6 = -14 + 2M$. The dual constraint associated with the non-basic variable x_{12} is violated. Thus $\bar{P}_{12}$ is selected as the entering vector.

Using the simplifications discussed at the end of section 5 and the procedure developed in [13] to find the representation of P_{12} yields

$$Z_{12} = \begin{pmatrix} \frac{1}{2} \\ -\frac{3}{4} \\ 0 \\ \frac{1}{4} \\ \frac{1}{16} \\ 0 \\ \frac{3}{4} \end{pmatrix}$$

The ratio test indicates that $\bar{P}_{y2}$ is the vector to leave the basis. Thus a case 1 basis exchange is performed. The basis graph of Figure 1 will not change except for the variable values. The new values (found by a standard pivot operation) are $x_{13} = 7$, $x_{24} = 2$, $x_{25} = \frac{5}{2}$, $x_{23} = 1$, $x_{34} = \frac{3}{2}$, $x_{46} = \frac{5}{3}$, and $x_{12} = 1$.

Since the representation of P_{12} is known in terms of B_n, the calculation of S is simplified. Setting $\theta = 1/(\frac{1}{4} - 1) = -\frac{4}{3}$ and $\rho = 1/2 - 1 = -1/2$ yields $S = 2/3$. Using the same procedure as discussed earlier, the dual variable values are found to be $w_1 = 9$, $w_2 = -23/3$, $w_3 = -12$, $w_4 = 44/3$,

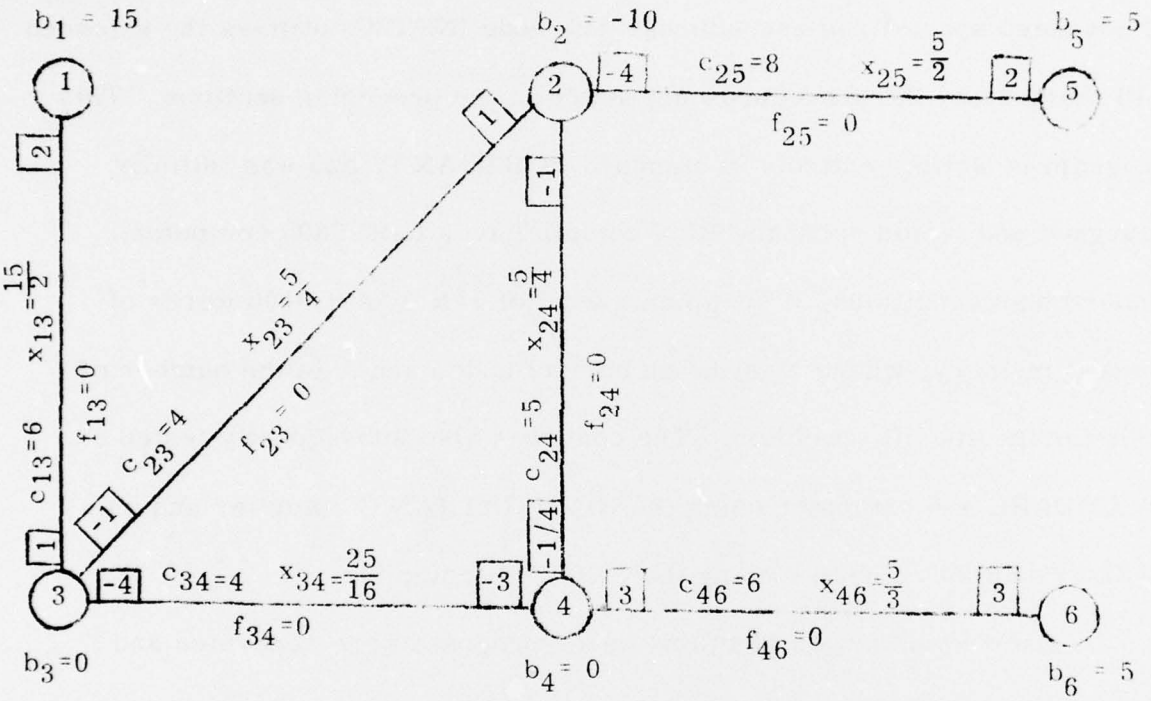


FIGURE 1

Initial Basis Graph

$w_5 = -40/3$, and $w_6 = -38/3$. This is a dual feasible solution, and therefore, the primal and dual solutions are optimal.

8.0 COMPUTATIONAL RESULTS

Code Development

A computer code was developed for solving singularly constrained generalized network problems in order to confirm the relative efficiency of advanced specialized procedures. The code (NETSG) utilizes the extended API method and the procedures presented in the preceding sections. This program is written entirely in standard FORTRAN IV and was initially debugged and tested using the RUN compiler on a CDC 6600 computer. Under those conditions, it occupies a total of 11N + 5A + 9100 words of central memory, where N is the number of nodes and A is the number of arcs for the specific problem. The code was also subsequently tested on an AMDAHL 470 computer using the IBM FORTRAN G compiler and on a CDC CYBER 74 computer using the CDC FTN compiler.

Each of the major functions of the program were separated and placed in subroutines. This made debugging easier and allowed for the use of timing procedures for each task. Such considerations are important if alternative procedures or parametric values are to be tested. The two decision rules to be considered in this paper are candidate list size and Big-M values.

One of the major considerations in any simplex based computer code is the choice of a vector to enter the basis on each iteration. One of the most efficient procedures for achieving this is the use of candidate lists (or multiple pricing). Candidate lists decrease the amount of time required to search for the "best" pivot arc by providing a selection list that can be

accessed after each iteration. A more complete description of an S - R candidate list is as follows:

> Examine sequentially each of the vertices for the underlying generalized network. For each vertex, select the non-basic edge that violates dual feasibility the most (20). Add this edge (if one exists) to the list and proceed to the next vertex. This is done until the list contains R entries or until all of the vertices have been explored. From the list the "best" edge is selected to enter the basis. The list is accessed after each pivot until it is void of eligible edges or until it has been accessed S times. At such time, the list is refilled by again examining each vertex, starting at the last examined vertex. When the last (largest numerical designation) is encountered, the non-graph dual constraints (9) are checked before proceeding to the first vertex. If a complete pass through the vertices is made without filling the list, the size of R is reset to the number found and S is reduced to 1/2R.

This type of procedure has been found to be exceptionally efficient for pure and generalized network codes. For this reason, the effect of the initial values of S and R for the singularly constrained code will be shown subsequently.

NETSG currently implements an artificial start procedure. In other words, the basis for the underlying generalized network is composed entirely of self-loops with Big-M cost coefficients. This method, although probably not the most efficient, provides a very quick initial basis. However, the magnitude of Big-M must be selected. This can be a very critical determination both in terms of total solution time and total number of pivots performed.

Choice of Decision Rules

The choice of the correct candidate list size and the magnitude of Big-M were both obtained by performing extensive computational tests. This testing required the use of a program to generate problem sets. Klingman, Napier, and Stutz have created the program NETGEN for generating pure network problems [19]. With NETGEN, the user can specify the desired structure of a problem and it will be generated using a random function. For test purposes, NETGEN was modified to create generalized arcs (non-unity multipliers) and coefficients for the extra constraint. Both of these are created using ranges specified by the user. The input specifications for the first three test problems can be found in table 1. All of these problems have 1000 nodes, but they differ dramatically in both the structure of the underlying generalized network and of the extra constraint.

The first test was to determine the proper initial size of the candidate list. The results of this test can be found in table 2. The sizes tested were 1-1, 5-10, 8-16, and 15-25. In all three of the problems, the 8-16 list was clearly the best choice. On problem 2, the 8-16 list proved to be over twice as fast as the 1-1 list procedure.

The next decision rule to be tested was the magnitude of Big-M. In the previous test, this value had been set to 1000. Additional values tested were 500, 250, 150, and 100. Each of these were coupled with an 8-16 candidate list. The results can be seen in table 3. In problems 1 and 3, the 150 value proved to be the best choice. In problem 2, the 250 value yielded the best total time but the 150 value gave the lowest number of total

TABLE 1

PROBLEM SPECIFICATIONS

Problem	Nodes	Sources	Sinks	Arcs	Transshipments Sources	Transshipments Sinks	Multiplier Range Min Max	Cost Range Min Max	Bound Range Min Max	Total Supply	Random Number Seed
1	1000	500	500	5000	0	0	.5 - 1.5	1 - 100	---	100000	13502460
2	1000	50	500	6000	5	100	.5 - 1.5	1 - 100	---	100000	13502460
3	1000	100	600	6000	25	100	.5 - 1.5	1 - 100	---	100000	13502460

	Extra Constraint			
	Non-Zero Coefficients	Coefficient Range Min	Max	Right-Hand Side Value
1	3000	-1.5	1.5	50000
2	4000	-2.0	2.0	20000
3	6000	-1.0	1.0	10000

TABLE 2

CANDIDATE LIST SIZE*

Problem	1-1		5-10		8-16		15-25	
	Total Time	Total Pivots	Total Time	Total Pivots	Total Time	Total Pivots	Total Time	Total Pivots
1	172.6	7095	119.7	5151	113.8	5031	134.4	5522
2	156.7	7747	82.8	4549	76.6	3577	88.5	3531
3	154.8	7255	103.1	4830	92.7	4591	109.3	4972

* times listed are in CPU seconds.

TABLE 3

BIG-M VALUES*

Problem	1000		500		250		150		100	
	Total Time	Total Pivots	Total Time	Total Pivots	Total Time	Total Pivots	Total Time	Total Pivots	Total Time	Total Pivots
1	113.8	5031	104.3	4612	92.4	4262	76.1	3705	51.9[1]	3018
2	76.6	3577	77.5	3502	48.9	2660	52.4	2566	33.1[2]	2341
3	92.7	4591	82.5	3524	76.3	3965	50.2	2832	39.9[3]	2333

*times listed are CPU seconds

[1]infeasible - one basic artificial

[2]infeasible - five basic artificials

[3]infeasible - one basic artificial

pivots. Note that the 100 value (equal the largest cost) proved to yield infeasible solutions in all three cases.

Code Comparisons

The next phase of testing was to compare the singularly constrained code NETSG with two widely used standard linear programming packages. In this case, the five problems indicated in table 4 were used as test data. These problems range from a 50 x 50 constrained generalized transportation problem to a 1000 node constrained transshipment problem.

The first comparison was made against the IBM MPS-360 linear programming package. The tests were performed on an AMDAHL 470 computer. Since both IBM-360 and AMDAHL 470 computers have smaller word sizes than a CDC 6600, NETSG would not initially run due to precision errors. Thus all type real variables were changed to double precision. The effect of this was two-fold. First, it greatly increased the amount of central memory required to run the program. Second, the use of double precision arithmetic routines significantly increased computation time.

The results of testing against MPS-360 can be found in Table 5. Even with the disadvantages listed above, NETSG proved to be between 5 and 8 times faster than MPS-360. The efficiency of NETSG was much more apparent as the problem size increased.

The last comparison was performed using the APEX-III linear programming package. This program is distributed by Control Data Corporation and the tests were conducted on a CDC CYBER-74 computer. In this case, there were no problems with word size or precision. The

basis of comparison, however, is not CPU seconds, but rather a CDC accounting measure called an SBU. The total SBU count for a job is an accumulation of CPU seconds used, I/O functions performed, and central memory requirements. This measure seemed appropriate, especially since it allowed the comparison of actual dollar amounts. CDC charges customers a minimum of $0.18 per SBU used.

The results of the comparison of NETSG and APEX-III are contained in Table 6. In this case, NETSG was from 6 to 28 times more efficient, depending upon the problem size. It should be noted that in problem 5 APEX-III used over 20 times as many SBU's and was still not at the optimum after completing 10,000 iterations.

TABLE 4

PROBLEM SPECIFICATIONS

Problem	Nodes	Sources	Sinks	Arcs	Transshipments Sources	Transshipments Sinks	Multiplier Range Min Max	Cost Range Min Max	Bound Range Min Max	Total Supply	Random Number Seed
1	100	50	50	1000	0	0	.5-1.5	1-100	---	100000	13502460
2	200	100	100	2000	0	0	.5-1.5	1-100	---	100000	13502460
3	200	50	100	1000	10	20	.5-1.5	1-100	---	100000	13502460
4	500	50	300	4000	5	50	.5-1.5	1-100	---	100000	13502460
5	1000	100	600	6000	25	100	.5-1.5	1-100	---	100000	13502460

	Non-zero Coefficients	Coefficient Range Min	Coefficient Range Max	Right-Hand Side Value
1	500	.5	1.5	750000
2	1500	.5	1.5	250000
3	250	.5	1.5	75000
4	2000	.5	1.5	500000
5	3000	.5	1.5	500000

TABLE 5

NETSG vs. MPS-360*

Problem	NETSG	MPS-360
1	1.64	8.59
2	3.92	31.27
3	2.61	16.48
4	13.06	110.68
5	35.62	DNR[1]

*times listed are CPU seconds

[1]problem data set was too large

TABLE 6

NETSG vs. APEX - III

Problem	NETSG		APEX-III	
	SBU's[1]	Cost[2]	SBU's	Cost
1	9.16	$ 1.65	62.70	$ 11.29
2	16.10	2.90	205.87	37.06
3	11.38	2.05	130.18	23.43
4	32.72	5.89	943.25	169.79
5	83.13	14.96	1875.55[3]	337.60

[1]System Billing Unit

[2]Computed at $0.18 per SBU

[3]Terminated after 10,000 iterations
Objective function value = 379,065,627
Optimal value = 4,745,739

Bibliography

[1] E. Balas and P. Ivanescu (Hammer), "On the Generalized Transportation Problem", Management Science, 11, 1(1964), 188-202.

[2] G. Bradley, G. Brown, and G. Graves, "A Comparison of Storage Structure for Primal Network Codes", Presented at the Joint ORSA/TIMS Conference, Chicago, April, 1975.

[3] A. Charnes and W. W. Cooper, Management Models and Industrial Applications of Linear Programming, Vols. I and II, Wiley, New York, 1961.

[4] R. Crum, D. Karney, and D. Klingman, "Large Scale Models of Multinational Companies", Final Report on OASIA Contract Number TOS-75-29, March 1976.

[5] G. Dantzig, Linear Programming and Extensions, Princeton University Press, Princeton, New Jersey, 1963.

[6] N. Deo, Graph Theory with Applications to Engineering and Computer Science, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1974.

[7] F. Glover, D. Karney, and D. Klingman, "The Augmented Predecessor Index Method for Locating Stepping Stone Paths and Assigning Dual Prices in Distribution Problems", Transportation Science, 6, 2 (1972), 171-179.

[8] F. Glover, D. Karney, and D. Klingman, "Implementation and Computational Comparisons of Primal, Dual, and Primal-Dual Computer Codes for Minimum Cost Network Flow Problems", Networks, 4, 3(1974), 191-212.

[9] F. Glover, D. Karney, D. Klingman, and A. Napier, "A Computational Study on Start Procedures, Basis Change Criteria, and Solution Algorithms for Transportation Problems", Management Science, 20, 5(1974), 793-813.

[10] F. Glover, D. Karney, D. Klingman, and R. Russell, "Solving Singularly Constrained Transshipment Problems", Research Report CCS 212, Center for Cybernetic Studies, University of Texas at Austin, 1974.

[11] F. Glover, and D. Klingman, "On the Equivalence of Some Generalized Network Problems to Pure Network Problems", Mathematical Programming, 4, 3(1973), 369-378.

[12] F. Glover and D. Klingman, "A Note on Computational Simplifications in Solving Generalized Transportation Problems", Transportation Science, 7, 4(1973), 351-361.

[13] F. Glover, D. Klingman, and J. Stutz, "Extensions of the Augmented Predecessor Index Method to Generalized Network Problems", Transportation Science, 7, 4(1973), 377-384.

[14] F. Glover, D. Klingman, and J. Stutz, "Implementation and Computational Study of a Generalized Network Code", Presentation at the 44th National ORSA Conference, San Diego, California, 1973.

[15] F. Glover, D. Klingman, and J. Stutz, "The Augmented Threaded Index Method for Network Optimization", INFOR, 12, 3(1974), 273-298.

[16] E. Johnson, "Programming in Networks and Graphs", ORC Report 65-1, University of California at Berkeley, 1965.

[17] E. Johnson, "Networks and Basic Solutions", Operations Research, 14, 4(1966), 619-623.

[18] D. Karney and D. Klingman, "Implementation and Computational Study on an In-Core Out-of-Core Primal Network Code", To appear in Operations Research, 24(1976).

[19] D. Klingman, A. Napier, and J. Stutz, "NETGEN - A Program for Generating Large Scale (Un) Capacitated Assignment, Transportation, and Minimum Cost Flow Network Problems", Management Science, 20, 5(1974), 814-821.

[20] D. Klingman and R. Russell, "On Solving Constrained Transportation Problems", Operations Research, 23, 1(1975), 91-107.

[21] R. Langley, "Continuous and Integer Generalized Flow Problems", Unpublished Dissertation, Georgia Institute of Technology, 1973.

[22] J. Maurras, "Optimization of the Flow Through Networks with Gains", Mathematical Programming, 3, (1972), 135-144.

[23] J. Mulvey, "Column Weighting Factors and Other Enhancements to the Augmented Threaded Index Method for Network Optimization", Presented at the Joint ORSA/TIMS Conference, San Juan, Puerto Rico, 1974.

[24] V. Srinivasan and G. Thompson, "Accelerated Algorithms for Labeling and Relabeling of Trees with Applications for Distribution Problems", JACM, 19, 4(1972), 712-726.

[25] V. Srinivasan and G. Thompson, "Benefit-Cost Analysis of Coding Techniques for the Primal Transportation Algorithm", JACM, 20 (1973), 194-213.